unit VI Visible Surface Determination

Defination

- Given a collection of 3D objects and a 2D viewing specification, the process of determining the lines or surfaces of 3D object that are visible and displaying only those surfaces or lines that are visible is called as visible surface determination.
- The process is also called as hidden line elimination & hidden surface elimination algorithm

Visible surface determination algorithm works on two approaches

1) object space method
2) Image space method.

1) object space method → visible surface determination is done on physical coordinates before projection.

2) Image space method: visible surface determination is done on screen coordinates after projection

Techniques for efficient visible surface determination

The techniques are basically meant for decreasing the overhead of the algorithm.

1. prespective transformation

- It is used to reduce the overhead involved in prespective projection visible surface determination.
- The technique aims on finding the visibility of the surface before it is projected on to the

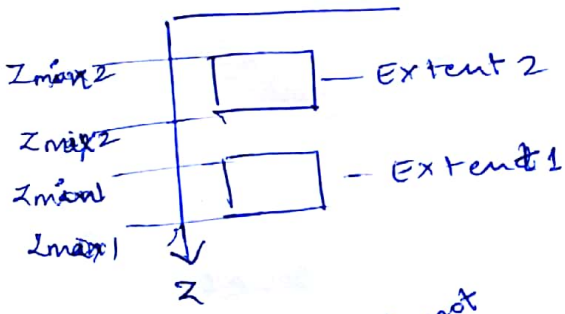
screen and as after ~~projection~~ projection loses some information on the depth.

- A 3D object is initially transformed such a way that parallel projection of ~~un~~transformed object is equals to the prespective projection of untransformed object.

2. Extends and bounding Box.

- This technique aims in the reduction of unnecessary compalisions between the objects for the visible surface determination.

- Extends are the rectangular surface surrounding the projection of the 3D object.



no overlap
if,

$z_{max2} < z_{min1}$

or

$z_{max1} < z_{min2}$

[If extends ~~overlap~~ donot then the objects donot overlap
If extends overlap objects may or may not overlap.]

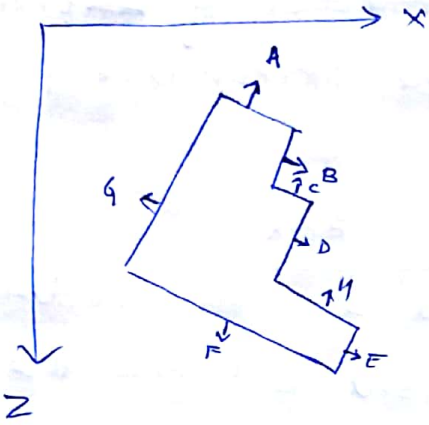
Bounding volumes: Rectangular surface enclosing the object itself is called an Bounding volume or a Bounding Box.

3. Back-face Culling

- A back face culling technique is applicable when the object is approximated as polyhedron whose interior is not exposed to the front side.
(polyhedron - objects made out of n number of polygons)

- The technique identifies those polygons whose surface normal points away from the observer and eliminates the surfaces out of the visible surface determination.

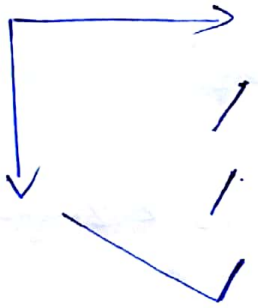
- Ex:



eliminate: A, C, ~~H~~ G

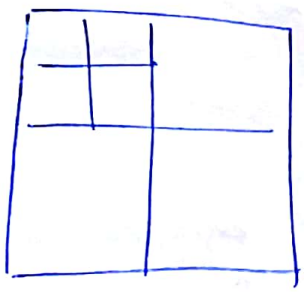
non eliminate: B, D, E, F

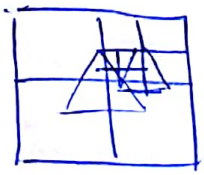
(Remove all eliminate surfaces)



4. Spatial Partition

- It is based on the divide and conquer strategy in which an area subdivision rule is applied to divide an entire area into smaller sub problems.
- Each area is examined to find out the visibility seperately and then the solutions are combined to get the final result.



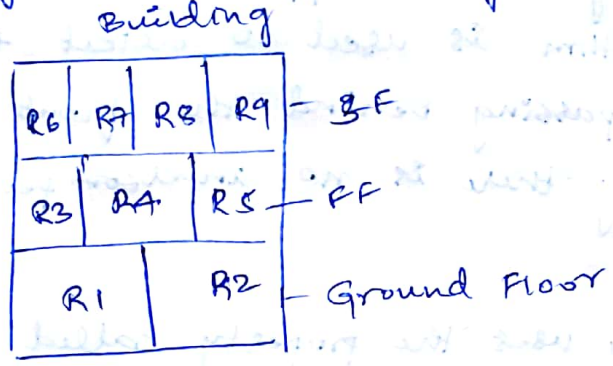


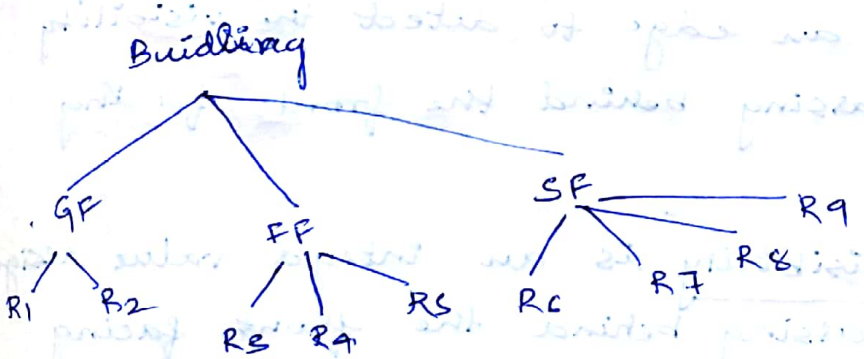
If there is overlap keep dividing the space until you get a solution

5. Heirarchy

A problem is basically realized in terms of hierarchical tree structure.

- In computer graphics an object given than needs to be projected it first realised as hierarchical tree structure and then visibility is determined individually in each hierarchy.





Algorithm for visible Line determination

1) Robert's algorithm
2) Apple's algorithm
3) Haloed lines.

1) Robert's algorithm

- Invented by Robert's.
- algorithm is only applicable for the convex polyhedron made out of multiple polygon surfaces.

Works in two phases

phase 1: the algorithm identifies the edges shared by the back facing polygons and eliminates

all those edges by the method of back face culling

phase 2: It compares each remaining ~~etq~~ edge with the other polyhedron to determine the visibility.

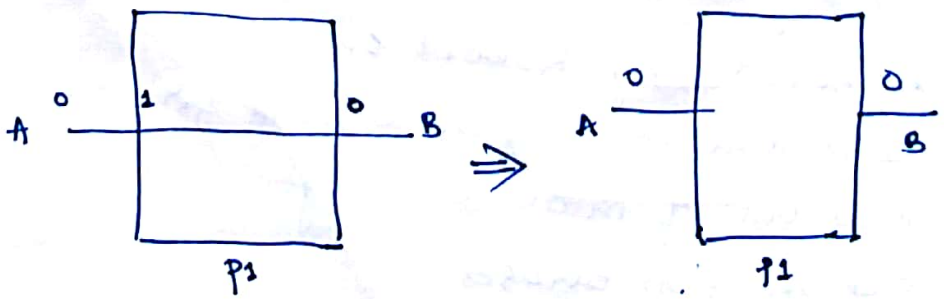
imp

2) Apples algorithm:

- invented by Arthur Apple
- The algorithm is used to detect the visibility of an edge passing behind the front facing polygon considering there is no interconn penetration of polygons.

- It basically uses the property called as Quantitative invisibility for an edge to detect the visibility for an edge passing behind the front facing polygon.

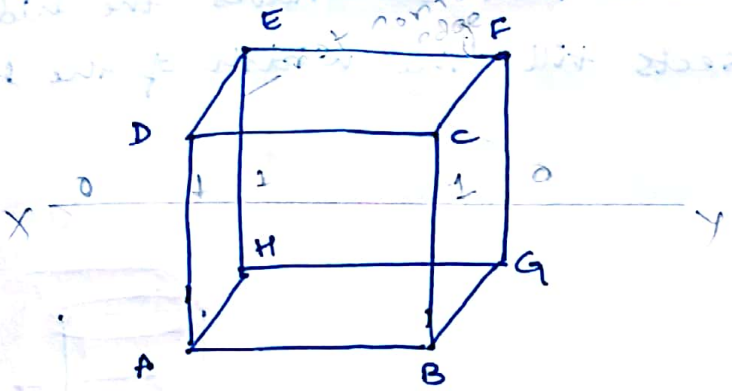
- Quantitative invisibility is an integer value assigned to an edge passing behind the front facing polygon and the integer factor is incremented by one when the edge enters inside the polygon and is decremented by 1 when the edge comes out of the polygon. A line or a part of the line is visible only when Quantitative invisibility is zero



(only that part where integer factor is 0 is visible)

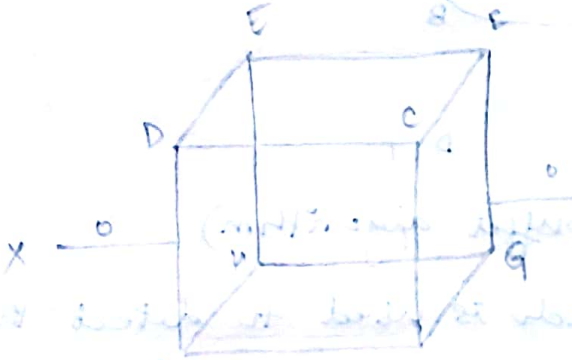
- considering the fact of non-interpenetrating polygons. The visibility of any edge passing through front facing polygon changes only when it passes behind the contour edge.

contour edge is an edge shared by front facing polygon and a back facing polygon or an unshared edge of front facing polygon.

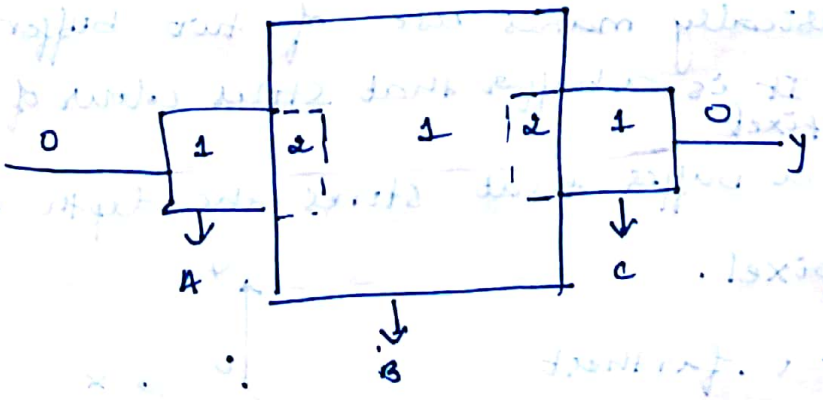


ABCD - front face.

contour edge: AB, AD, GF, EF, BG, DE

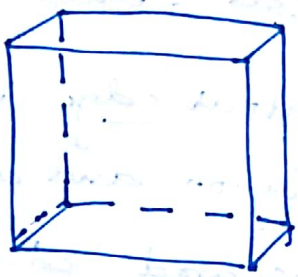


eg:



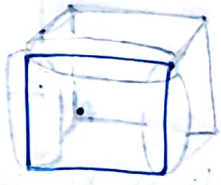
③ Haloed Lines

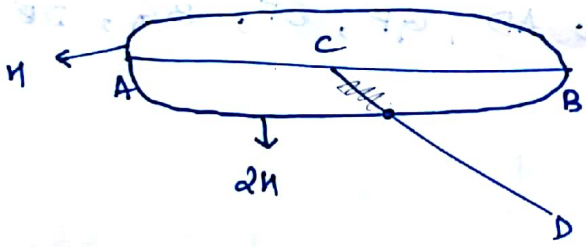
- The algorithm is used to display hidden lines in the form of dots and dashes.



- The algorithm encloses the visible edges by an hallow of radius H and width 2H and then eliminates the hidden lines that intersects till the width of the hallow.

eg: CD - hidden
AB - visible



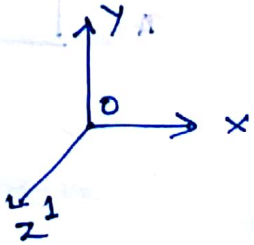


Z Buffer Algorithm (depth buffer algorithm)

- An image space method which is used to detect the visible surfaces and display only those surfaces that are visible.

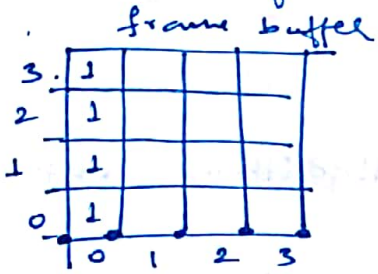
- The algorithm basically makes use of two buffers
  1. Frame buffer : It is a buffer that stores colour of each pixel
  2. Z buffer: It is a buffer that stores the depth value of each pixel.

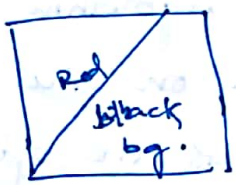
depth - 0 - farthest
1 - nearest



Assume there is a screen of 5×5

So there is a frame buffer of 5×5





Algorithm

```
for(y=0; y<ymax; y++)
  {
    for(x=0, x<xmax; x++)
        {
          z[x][y] = 0;
          F[x][y] = BG_COLOR;
        }
  }
For (each polygon in the list)
  {
  for( y=ymin; y<=ymax; y++)
    {
     for(x=xmin, x<=xmax; x++)
       {
    d= (depth) calculate dipth at pixel (x,y);
      if(d>=z[x][y])
            {
             z[x][y] = d;
             F[x][y] = POLYGON_COLOUR;
    } } } }
```

List priority algorithm:

They create a list of objects on some priority criteria

1) Depth sort algorithm:
2) Binary Space Partitioning Tree Algorithm (BSP)

imp
1) Depth sort algorithm (Painter's algorithm)

- It is an algorithm in which objects are given the priority based on the depth value.
- Farther objects are given the highest priority than the closer objects ie the objects are sorted initially based on the decreasing depth value ie they are sorted on the lowest z coordinate value.

P
one selected to paint

Q
whatever is remaining in the list (is not going to obscure any)

we have to see that P never overlays Q

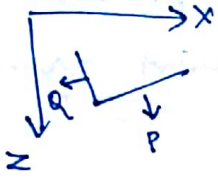
Fa.

Conceptual steps

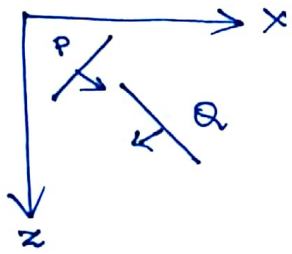
1. Sort all the polygons based on the increasing value of z coordinates
2. For the selected polygone P from the list check if P obscures any Q remaining in the list and if P obscures Q then resolve the ambiguity, else goto step 3.
3. Scan convert polygon P

Sequential test to prove P does not obscure Q

1. If z extents of P do not overlap z extents of Q
2. If x extents of P do not overlap x extents of Q
3. If y extent of P do not overlap y extent of Q
4. If P is completely on the opposite side of Q's plane as view point.



5. If Q is completely on the same side of P's plane as viewpoint



Algorithm

1: Sort the list of polygons based on the smallest z coordinate value and create a sorted list called polylist.

2: Select the ~~first~~ ~~polygon~~ and remove the first polygon from the ~~f~~ polylist and name it as P.

3: For each polygon Q in polylist do the following steps.

i) Check if every Q passes test 1 to 5 with respect to P

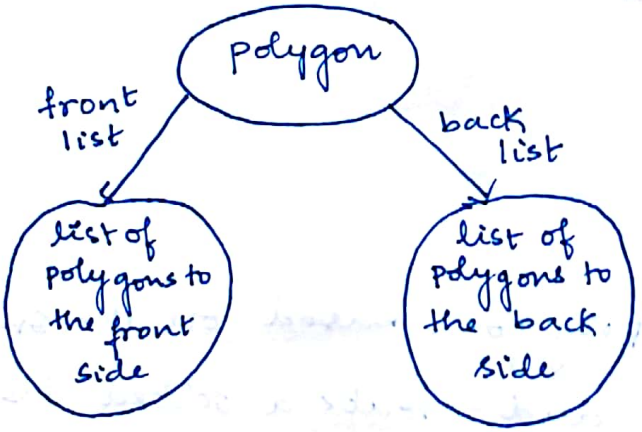
ii) If any Q fails the test then swap P and Q and split polygon Q into Q1 and Q2. with respect to P's plane and replace Q1 and Q2 with Q and goto step 1. else goto step iii

iii) Paint the polygon ?

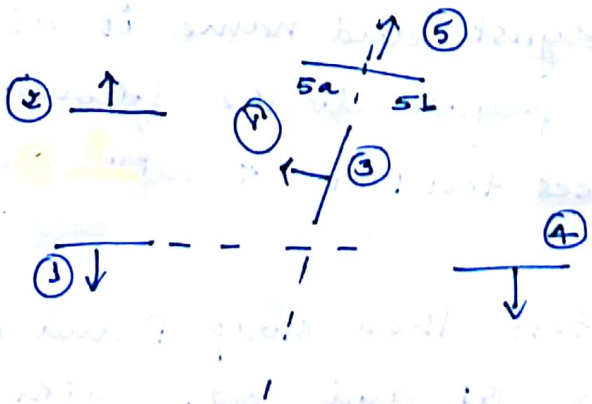
4. Goto step 2.

Binary space Partitioning (BSP) Tree algorithm

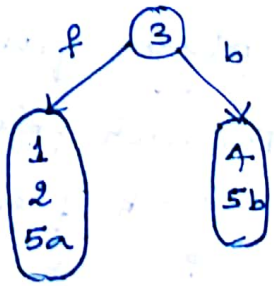
- In this algorithm list is created based on the binary tree structure.
- This algorithm doesnot require any presorting process and a tree structure is created arbitarily from a particular polygon.
- This algorithm is even useful in creating a display from any viewpoint.



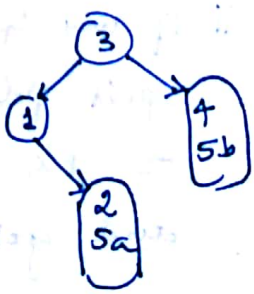
eg:



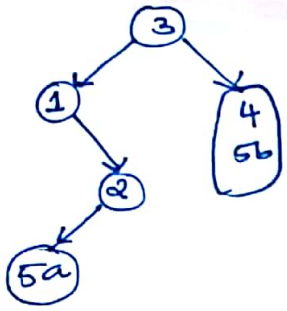
(i)



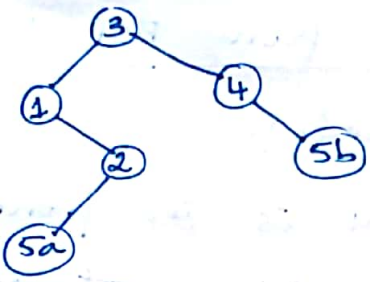
(ii)



(iii)



iv)



imp algorithm : BSP tree algorithm

```
Create_BSP_TREE (polylist)
{
 if (root == NULL)
   {
    return;
    }
root = select and remove a polygon from polylist
for (each remaining polygon p in the polylist)
   {
    if(p in front of root)
      {
       add_BSP (p, root, frontlist);
       }
    else if(p in back of root)
       {
       add_BSP(p, root, backlist);
       }
    else
      {
       split_polygon(p, root, frontpart, backpart);
       add_BSP(frontpart, root, frontlist);
       add_BSP( backpart, root, backlist);
      }
   }
Create_BSP_Tree (frontlist);
Create_BSP_Tree (backlist);
}
```

Displaying list from BSP tree

Recursive proceedure

1) If viewpoint is in front of root
   (i) Recursively trace root → backchild
   (ii) visit root
   (iii) Recursively trace root → frontchild

2) If viewpoint is towards back of root
   (i) Recursively trace root → frontchild
   (ii) visit root
   (iii) Recursively trace root → backchild

4, 5b, 3, 1, 2, 5a } → For previous diagram.

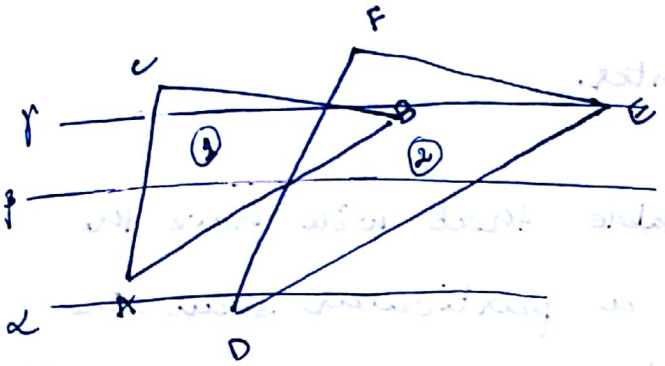
Pseudocode:

```
display (root)
    {
      if (root == NULL)
        {
         return
        }
      else (v in front of root)
         {
         display (root → backlist)
         visit (root);
         display (root → frontlist)
         }
      else
         {
         display (root → frontlist)
         visit (root);
```

```
    display(root → backlist)
    }
}
```

scanline algorithm

- Similar to scanline algorithm for polygon filling.
- deals with multiple polygon surfaces.

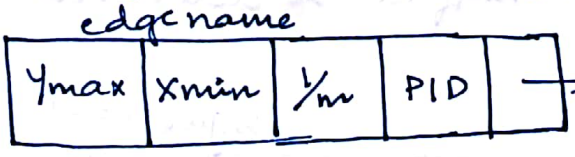


This algorithm uses 3 data structures

1. Edge Table
2. Active edge Table
3. Polygon surface table

Edge Table :- This table stores all the information regarding the edges that are intersected by each scan line, $y = y_{min}$ to y_{max}

- Each edge information is stored in the form of a node having 5 different edges.



(points to the next edge of same scanline)

Y_{max} - largest y coordinate value for the edge

X_{min} - Smallest x coordinate value for the edge

m - slope

PID - ID of a polygon surface to which the edge belongs to.

pointer - pointing to next edge of the same scan line.

eg:- Consider the previous diagram.

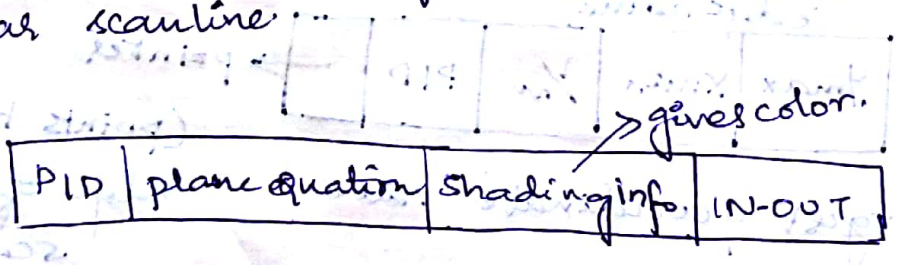
for edge AC

Active edge table :- It is a table that will have the list of edges activated for a particular scan line that ranges from y=ymin to ymax

Ex : Structure of AET

~~imp~~ Scanline	~~Arbi~~ EdgeList
α	DF, DE
β	AC, ~~AB~~, DF, AB. DE
γ	AC, BC, DF, EF

Polygon surface table :- It is a table that keeps track of the information of each polygon surface and it also tells if a particular surface is activated for a particular scanline.



plane equation → used to derive depth

IN-OUT → boolean field which specifies if the surface is active or not.

0 - NOT active 1 - Active

eg: polygon surface table

1	PE1	Blue	0
2	PE2	Redr	0

Assume polygon surface 2 is in front of 1

for scanline $\alpha \longrightarrow$ Red.

β

Algorithm

1. Initialize ET
2. Initialize polygon surface table
3. $y = y_{min}$ smallest y coordinate value.
4. for $y = y_{min}$ to y_{max} do the following.
   i) $x = x_{min}$ of intersecting edge.
   ii) invert IN-OUT flag corresponding to the intersected edge
   iii) If $y = y_{max}$ of an edge then delete the edge from AET
   iv) count = number of IN-OUT flags that are 1
   v) If (count > 1) then find the ~~closed~~ closest surface and color the pixel range with shade of the closest surface
      else
      goto vi
   vi) Shade the range of pixel with the colour of activated surface
   vii) $x = x + 1/m$
   viii) $y = y + 1$

Area subdivision Algorithm

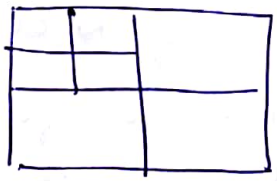
- works based on the rule of divide and conquer strategy used in spatial partitioning.
- It basically tries to make the decision on the ~~rea~~ area of projection and if the decision couldn't be made then the area is recursively divided into smaller part.

2 algorithm

1. Warnock's algorithm
2. Weiler Atherton algorithm

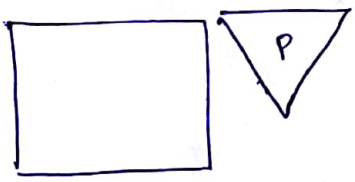
Warnock's algorithm

- Initially the algorithm treats entire area of the projection to be area of interest and finds ~~of~~ out the relationship of each sorted polygon with respect to area of interest and if it couldn't make the decision based on the relationship derived then entire area is recursively subdivided into 4 equal parts of rectangular region and it continues to make the decision.



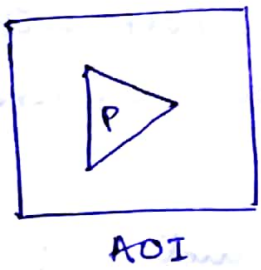
Four types of relationship with respect to area of interest

i) Disjoint polygon: when a particular polygon is completely out the area of interest (AOI)



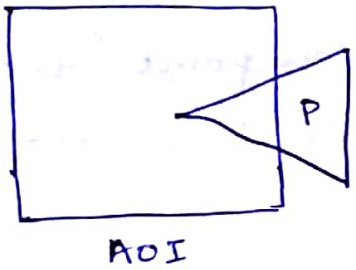
⇒ P is a disjoint polygon W.R.T to AOI

2) contained polygon: when a polygon is completely inside the AOI



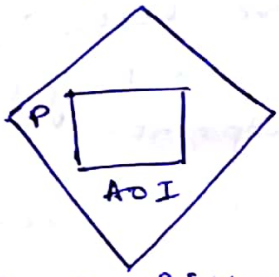
⇒ P is contained w.r.t AOI

3) Intersecting polygon: when a part of polygon is inside and a part of polygon is outside



⇒ P is intersecting w.r.t AOI

4) Surrounding polygon: when a polygon surrounds the AOI.



⇒ P is surrounding AOI

imp

Warnocks algorithm

step 1: Initialize entire screen to be area of interest

step 2: Create a list of polygons sorted in some order based on the z coordinate value with respect to area of interest.

step 3: For each polygon in the list derive the relationship with respect to area of interest

step 4: Perform the following test to make the decision

- test 1: If all the polygons in the list are disjoint wrt area of interest then color the area of interest with background color.

test 2: If there is a single contained or intersecting polygon wrt the area of interest then first color the area of interest with background color and then fill the area acquired by the polygon with polygon colour.

test 3: If there is a single surrounding polygon then color the AOI with polygon color.

test 4: If there are multiple polygons in area of interest and if there is a single surrounding polygon closer to the viewpoint then color the area of interest with surrounding polygon color.

test 5: If the area of interest is a single pixel P(x,y) and if all the above test's fails then the area is coloured with the polygon color that is closest to the viewpoint.

Step 5: If all the test fails then recursively divide the area of interest into 4 equal rectangular parts and go to step 2.

imp

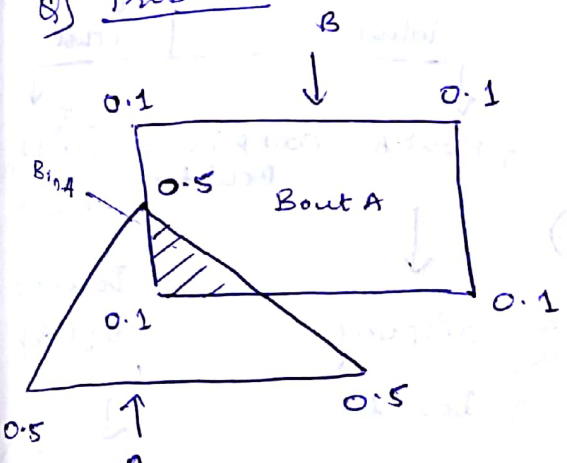
Weiler Atherton algorithm

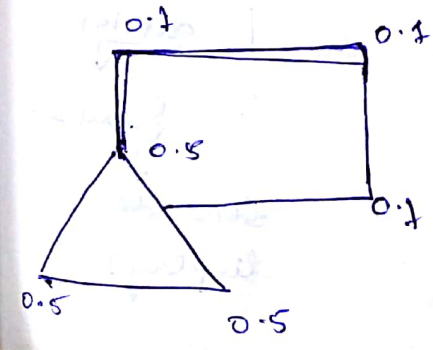
- The algorithm divides the area of projection based on the boundary of a particular polygon called as clip polygon rather than dividing it based on the rectangular region like warnock's algorithm.
- clip polygon is the closest polygon that comes in the list of polygons.

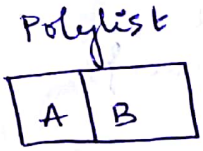
Conceptual steps

1. sort the list of polygons based on the decreasing value of z coordinates.
2. select the first polygon from the list to be the clip polygon.
3. With respect to the clip polygon subdivide the list into in list and out list even considering the clip polygon itself.
4. In the inlist if any polygon P is farther from the clip polygon then delete those polygons as they are hidden.
5. If any polygon in the inlist is closer than the clip polygon then subdivide the inlist for each closer polygons recursively.
6. If inlist has only the clip polygon then display it.
7. Apply the steps recursively for the outlist.

Q] Problem







Subdivide (A {B})

inlist | outlist

{A, BinA}

BinA is completely farther from A so delete it.

{A}

displayA

{BoutA}

apply Subdivide (BoutA, { })

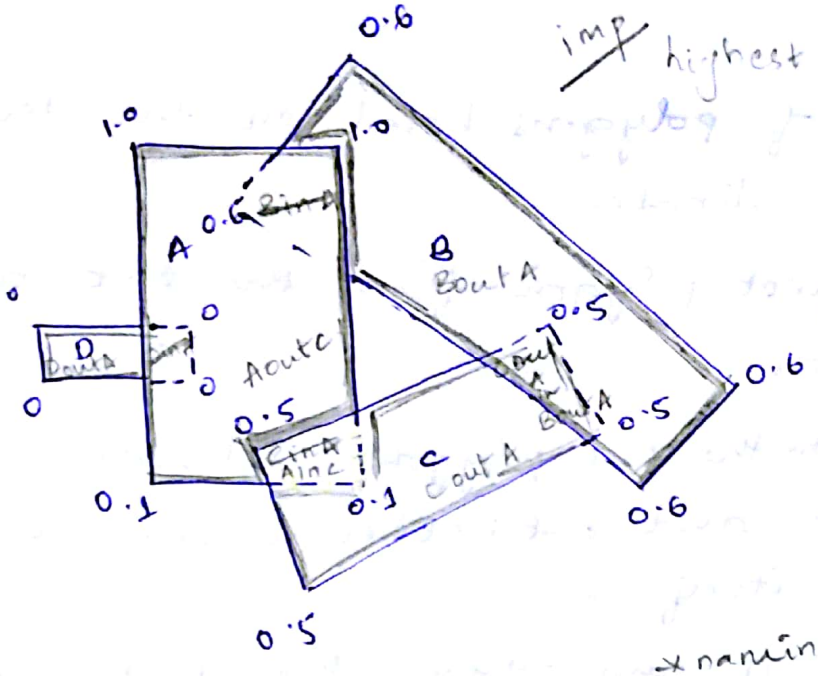
inlist | outlist

{BoutA}

{ }

display Bout A.

9]



imp highest is up

* naming can be anyways.

Polylist

| A | B | C | D |

↓

subdivide (A, {B, C, D})

inlist | outlist

{A, ~~BinA~~, CinA, ~~DinA~~} → consider CinA as clip polygon.

{BoutA, CoutA, DoutA}

subdivide (CinA, A)

inlist | outlist

{CinA, ~~AinC~~} → display CinA

{Aoutc} → subdivide {Aoutc, {}}

inlist | outlist

{Aoutc} → display Aoutc

{}.

subdivide (BoutA, {CoutA, DoutA})

inlist | outlist

{BoutA, ~~CoutA~~ ~~BoutA~~} → display BoutA

{CoutA, DoutA, BoutA, DoutA, BoutA}

subdivide (I, {J})

inlist | outlist

{I} → display I

{J, I} → ~~subdivide~~ display J, I

call subdivide

Weiler Atherton Algorithm Pseudo code

```
WA_subdivide (clip polygon, polylist[I])
{
  if (polylist == NULL)
    {
    return;
    }
  else
  { int inlist = NULL;
    outlist = NULL;
(3) for (each polygon in polylist)
    {
      inlist = list of polygons that is inside
               the clip polygon
      outlist = list of polygons that is
                outside the clip polygon
    }
(4) for (each polygon in inlist)
    {
     delete polygon that is behind the
     clip polygon
    }
(5) for (each polygon in inlist that is not part
         of clip polygon)
    {
    WA_subdivide (polygon_inlist);
    }
  (6) display polygon in inlist;
      polylist = outlist;
  (2) clippolygon = first polygon (polylist);
      WA_subdivide (clippolygon, polylist);
}
```

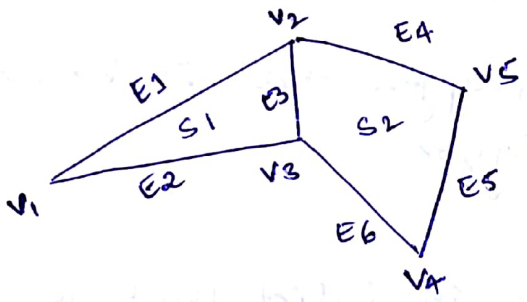
1) Polylist = sort polygons based on highest z coordinate
   clip polygon = first polygon (polylist);
2) WA_subdivide (clippolygon, polylist);

Unit III

Three-Dimensional Object Representation

Representation of polygon surfaces

- Usually in computer graphics many objects are represented in terms of combination of the polygon surfaces.
- The reason for this is the polygon surfaces are easy to render on the screen since the polygon surface uses linear equation of the form $Ax + By + Cz + D = 0$ which is easy to process in terms of mathematical operations.



- To represent a polygon there are 3 geometric tables

1) vertex table
2) Edge table
3) polygon surface table.

vertex table : An vertex table for a polygon consists of the collection of vertices as index and within each index it holds the (x, y, z) coordinate values corresponding to a particular vertex.